

INSTITUTO FEDERAL DO ESPÍRITO SANTO  
CURSO SUPERIOR DE SISTEMAS DE INFORMAÇÃO

**JOÃO PEDRO GARCIA PEREIRA  
RODRIGO COUTO RODRIGUES**

**APRENDIZADO POR REFORÇO DO LUNAR LANDER E CAR RACING**

Serra  
2024

JOÃO PEDRO GARCIA PEREIRA  
RODRIGO COUTO RODRIGUES

## **APRENDIZADO POR REFORÇO DO LUNAR LANDER E CAR RACING**

Relatório apresentado ao professor da matéria “Inteligência Artificial” do Curso de Sistemas de Informação do Instituto Federal do Espírito Santo, Campus Serra, como requisito parcial para a obtenção de nota.

Professor: Sergio Nery Simoes

Serra  
2024

## LISTA DE FIGURAS

Figura 1 – Recompensa Inicial do Lunar . . . . .	10
Figura 2 – Recompensa Final do Lunar . . . . .	10
Figura 3 – Lunar 1 milhão . . . . .	10
Figura 4 – Video Lunar Lander (Clique aqui) . . . . .	11
Figura 5 – Recompensa Inicial do Car Racing . . . . .	12
Figura 6 – Recompensa Final do Car Racing . . . . .	12
Figura 7 – Video Car Racing 25 Mil steps (Clique aqui) . . . . .	13
Figura 8 – Video Car Racing 50 Mil steps (Clique aqui) . . . . .	13
Figura 9 – Video Car Racing 100 Mil steps (Clique aqui) . . . . .	13
Figura 10 – Video Car Racing 200 Mil steps (Clique aqui) . . . . .	14
Figura 11 – Video Car Racing 400 Mil steps (Clique aqui) . . . . .	14

## Lista de tabelas

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO . . . . .</b>	<b>4</b>
1.1	Conceitos principais . . . . .	5
<b>2</b>	<b>METODOLOGIA . . . . .</b>	<b>6</b>
2.1	Políticas utilizadas . . . . .	6
2.2	Parametros utilizados . . . . .	6
2.3	Lunar Lander . . . . .	8
2.4	Car Racing . . . . .	9
<b>3</b>	<b>RESULTADOS E DISCUSSÃO . . . . .</b>	<b>10</b>
3.1	Lunar Lander . . . . .	10
3.2	Car Racing . . . . .	12
<b>4</b>	<b>ALGORITMOS . . . . .</b>	<b>15</b>
4.1	Lunar Lander . . . . .	15
4.2	Car Racing . . . . .	18
	<b>REFERÊNCIAS . . . . .</b>	<b>21</b>

## 1 INTRODUÇÃO

O aprendizado por reforço (Reinforcement Learning, RL) é uma área do aprendizado de máquina inspirada em como agentes inteligentes tomam decisões em um ambiente para maximizar algum tipo de recompensa cumulativa. Diferente do aprendizado supervisionado, onde o modelo é treinado com um conjunto de dados rotulados, no aprendizado por reforço, o agente aprende a tomar decisões com base em interações com o ambiente. Através das interações com o ambiente, o agente recebe feedback na forma de recompensas positivas ou negativas, o que lhe permite aprender a executar ações que conduzem à solução de problemas de maneira eficaz.

Existem alguns exemplos notáveis de casos em que agentes de aprendizado por reforço superaram jogadores profissionais em jogos complexos

Em 2019, o AlphaStar, um agente de aprendizado por reforço, alcançou um nível de habilidade em StarCraft II, um jogo de estratégia em tempo real, que o colocou entre os melhores jogadores do mundo. Ele venceu vários jogadores profissionais em confrontos diretos. StarCraft II é um jogo particularmente desafiador para a IA devido à sua complexidade, necessidade de planejamento estratégico e tomada de decisões em tempo real (ARULKUMARAN; CULLY; TOGELIUS, 2019).

Nesse relatório demonstramos o aprendizado por reforço aplicado ao treinamento de agentes nos ambientes "Lunar Lander" e "Car Racing", disponíveis na biblioteca Gymnasium. Utilizando a biblioteca Stable Baselines, investigamos a aplicação de diferentes políticas de aprendizado por reforço para solucionar os problemas desses dois contextos distintos.

## 1.1 Conceitos principais

- Agente: É a entidade que toma as decisões no ambiente. O objetivo do agente é maximizar a recompensa acumulada ao longo do tempo.
- Ambiente: É tudo com o que o agente interage. O ambiente fornece ao agente informações sobre o estado atual e responde com recompensas baseadas nas ações que o agente realiza.
- Estado: É a representação atual da situação do ambiente em que o agente se encontra. Ele contém todas as informações necessárias para o agente tomar uma decisão.
- Ação: É a escolha que o agente faz em um dado estado. As ações alteram o estado do ambiente.
- Recompensa: É o feedback que o agente recebe do ambiente após tomar uma ação. A recompensa pode ser positiva ou negativa, dependendo do impacto da ação sobre o objetivo final.
- Política : É a estratégia usada pelo agente para decidir qual ação tomar em cada estado. A política pode ser determinística, onde uma ação específica é escolhida para cada estado, ou estocástica, onde a ação é escolhida com base em uma distribuição de probabilidades.
- Função de Valor : Esta função estima o quão bom é um estado (ou um par estado-ação) em termos de recompensa acumulada que pode ser obtida a partir daquele estado.
- Exploração vs. Exploração : Um dos desafios do aprendizado por reforço é o equilíbrio entre explorar novas ações para descobrir suas recompensas e explorar ações conhecidas que já fornecem uma boa recompensa.

## 2 METODOLOGIA

Para o treinamento dos agentes nos ambientes, utilizamos a biblioteca Stable-Baselines3 em conjunto com os ambientes disponíveis no Gymnasium.

Ambos modelos foram inicializados com o algoritmo PPO (Proximal Policy Optimization), que é conhecido por ser eficiente e estável para problemas de controle contínuo e discreto.

### 2.1 Políticas utilizadas

Antes de iniciar o treinamento de ambos ambientes, buscamos entender melhor como cada um funcionava, para assim descobrir qual seria a melhor política a ser utilizada em cada um dos casos. Para escolher a melhor política para os ambientes, é importante considerar a natureza dos dados de entrada de cada ambiente:

Lunar Lander apresenta estados definidos por uma combinação de características, como posição, velocidade, ângulos, entre outros. Essas características são representadas por um vetor de entrada relativamente pequeno, portanto escolhemos a MlpPolicy. Essa política usa uma rede neural totalmente conectada (MLP) que é bem adequada para lidar com estados que são representados como vetores. A MlpPolicy geralmente oferece um bom desempenho, pois lida eficientemente com os dados tabulares/vetoriais fornecidos pelo ambiente.

O Car Racing fornece uma entrada visual (imagens), onde cada frame é uma imagem RGB. Neste caso, é necessário extrair características relevantes das imagens para a tomada de decisão, por isso optamos pela CnnPolicy, essa política utiliza redes neurais convolucionais (CNNs), que são bem adaptadas para a extração de características de imagens. Como o Car Racing depende de uma entrada visual, a CnnPolicy é ideal para capturar as nuances das imagens e fornecer boas políticas de ação.

Se fosse o caso de um ambiente que requer a combinação de entradas visuais e de outro tipo (como vetores de estados), o MultiInputActorCriticPolicy poderia ser mais adequado. Mas, para os dois ambientes mencionados, as políticas mencionadas acima parecem ser as mais eficazes.

### 2.2 Parametros utilizados

verbose: Controla o nível de detalhes dos logs durante o treinamento. verbose=1 significa que informações básicas sobre o processo de treinamento serão exibidas. Ajuda a monitorar o progresso do treinamento sem sobrecarregar o usuário com muitos detalhes. O nível 1 fornece uma boa quantidade de feedback sem ser excessivamente detalhado.



`learning_rate`: Define a taxa de aprendizado do algoritmo, controlando o tamanho dos passos que o modelo dá na direção dos gradientes durante a otimização. O valor de  $3e-4$  é um valor padrão que oferece um equilíbrio entre convergência rápida e estabilidade. É pequeno o suficiente para evitar grandes saltos no espaço de soluções, o que pode evitar instabilidades no treinamento.

`n_steps`: Número de passos de ambiente coletados antes de realizar uma atualização de política. Um valor de 2048 é um compromisso entre coleta de dados suficiente para uma atualização eficaz e a frequência das atualizações, evitando tanto o underfitting quanto o overfitting.

`batch_size`: Define o tamanho do lote usado durante a otimização da política. Um batch size de 64 é um valor comum, suficientemente grande para estimativas estáveis de gradiente, mas pequeno o bastante para garantir uma atualização frequente da política.

`n_epochs`: Número de épocas para treinar o modelo a cada vez que uma atualização de política é realizada. Treinar por 10 épocas permite que o modelo aprenda a partir do lote coletado, sem sobrecarregar o treinamento ou superajustar ao lote atual.

`gamma`: Fator de desconto para recompensas futuras. Este parâmetro determina a importância das recompensas futuras em relação às imediatas. Um valor de 0.99 significa que o algoritmo valoriza as recompensas futuras, mas ainda dá alguma importância para as recompensas imediatas.

`gae_lambda`: Parâmetro que controla o uso do Generalized Advantage Estimation (GAE), uma técnica que suaviza a estimativa de vantagem. O valor de 0.95 é um bom compromisso entre viés e variância, proporcionando uma estimativa mais estável da vantagem, o que melhora o desempenho do agente.

`clip_range`: Controla a faixa de clipping no PPO, que restringe as atualizações de política para evitar mudanças drásticas que possam prejudicar a estabilidade do treinamento. O valor de 0.2 é uma escolha padrão que permite que a política evolua sem grandes saltos, promovendo uma convergência mais estável.

`ent_coef`: Coeficiente de entropia, que incentiva a política a explorar mais, evitando que ela se torne excessivamente determinística. O valor de 0.01 é um compromisso comum entre explorar novas ações e aproveitar as ações já aprendidas, ajudando a evitar que o modelo fique preso em um mínimo local.

### 2.3 Lunar Lander

A política escolhida foi a MlpPolicy, adequada para entradas que são vetores de características, como é o caso do Lunar Lander.

O ambiente foi configurado utilizando a API do Gymnasium (`gym.make("LunarLander-v2")`). Fornecendo ao agente um vetor de estado contínuo composto por 8 valores, representando a posição, velocidade, ângulo e velocidade angular do módulo lunar, além de dois valores booleanos indicando se as pernas do módulo estão em contato com o solo. As ações disponíveis são discretas, permitindo ao agente escolher entre não fazer nada, disparar o motor principal ou disparar os motores laterais.

```

1 model = PPO(
2     "MlpPolicy",
3     "LunarLander-v2",
4     verbose=1,
5     learning_rate=3e-4,
6     n_steps=2048,
7     batch_size=64,
8     n_epochs=10,
9     gamma=0.99,
10    gae_lambda=0.95,
11    clip_range=0.2,
12    ent_coef=0.01,)

```

Antes do treinamento, o agente foi avaliado em 10 episódios utilizando um ambiente separado para teste, resultando em uma recompensa média que reflete o comportamento aleatório inicial.

```

1 eval_env = gym.make("LunarLander-v2")
2 mean_reward, std_reward = evaluate_policy(
3     model,
4     eval_env,
5     n_eval_episodes=10,
6     deterministic=True,)
7 print(f"mean_reward={mean_reward:.2f} ± {std_reward}")

```

Durante o treinamento, o modelo ajusta sua política com base no retorno acumulado em cada episódio, buscando maximizar a recompensa total ao longo do tempo.

O treinamento é realizado em múltiplos episódios, onde o agente interage com o ambiente, atualizando seus parâmetros de acordo com o feedback obtido. A cada 2048 passos de interação, os parâmetros do modelo são ajustados, permitindo uma adaptação gradual à tarefa de pousar o módulo lunar. O treinamento foi conduzido por 1.000.000 timesteps, com atualizações frequentes dos parâmetros da política.

## 2.4 Car Racing

A política escolhida foi a CnnPolicy, adequada para entradas de imagens, como as observações visuais do ambiente Car Racing.

O ambiente foi configurado usando a API do Gymnasium (`gym.make("CarRacing-v2", render_mode="rgb_array")`), que fornece ao agente uma observação visual do estado do carro e da pista em forma de imagens RGB. A tarefa do agente é controlar o carro utilizando ações contínuas para acelerar, frear e girar o volante.

```

1 model = PPO(
2     "CnnPolicy",
3     env,
4     verbose=1,
5     learning_rate=3e-4,
6     n_steps=2048,
7     batch_size=256,
8     n_epochs=10,
9     gamma=0.99,
10    gae_lambda=0.95,
11    clip_range=0.2,
12    ent_coef=0.01,
13 )

```

Antes do treinamento, o agente foi avaliado em 5 episódios usando um ambiente separado de avaliação, resultando em uma recompensa média que reflete o comportamento aleatório inicial.

```

1 eval_env = gym.make("CarRacing-v2", render_mode="rgb_array")
2
3 mean_reward, std_reward = evaluate_policy(
4     model,
5     eval_env,
6     n_eval_episodes=5,
7     deterministic=False,
8 )
9
10 print(f"mean_reward={mean_reward:.2f} ± {std_reward}")

```

Durante o treinamento, o modelo ajusta sua política com base no retorno acumulado em cada episódio, buscando maximizar a recompensa total ao longo do tempo.

O treinamento foi conduzido por 25.000, 50.000, 100.000, 200.000 e 400.000 timesteps, com atualizações frequentes dos parâmetros da política.

### 3 RESULTADOS E DISCUSSÃO

#### 3.1 Lunar Lander

Após treinar o agente no ambiente Lunar Lander usando o algoritmo PPO, foi possível observar uma evolução no desempenho do agente ao longo do tempo. A avaliação inicial do agente, realizada antes do início do treinamento, resultou em uma recompensa média de aproximadamente -992.26, com uma variação de +/- 634.709, indicando que o agente não era capaz de realizar pousos controlados e seguros. Esse desempenho é esperado para um agente sem treinamento, que age de maneira aleatória e sem qualquer conhecimento prévio do ambiente.

Figura 1 – Recompensa Inicial do Lunar

```
mean_reward=-992.26 +/- 634.7097855788866
```

Fonte: Elaborado pelo Autor, 2024.

Após o treinamento, que consistiu em 1.001.472 timesteps e 489 iterações, o agente alcançou uma recompensa média final de 249.79, com uma variação de +/- 45.883. Este resultado demonstra uma melhora na habilidade do agente de controlar o módulo lunar e realizar pousos bem-sucedidos dentro da zona designada.

Figura 2 – Recompensa Final do Lunar

```
mean_reward=249.79 +/- 45.883170449193365
```

Fonte: Elaborado pelo Autor, 2024.

A perda de valor final foi de 66.8, enquanto a perda do gradiente da política foi pequena, em torno de -0.000399, sugerindo que a política estava próxima de um ótimo local na função de recompensa.

Figura 3 – Lunar 1 milhão

```
-----
| rollout/                |          |
|   ep_len_mean           |         353          |
|   ep_rew_mean           |         248          |
| time/                   |          |
|   fps                   |         428          |
|   iterations            |         489          |
|   time_elapsed          |         2336         |
|   total_timesteps       |      1001472         |
| train/                  |          |
|   approx_kl             |    0.0044199713      |
|   clip_fraction         |    0.0371            |
|   clip_range            |    0.2               |
|   entropy_loss          |    -0.221            |
|   explained_variance     |    0.743             |
|   learning_rate         |    0.0003            |
|   loss                  |    21.5              |
|   n_updates             |    9770              |
|   policy_gradient_loss  |   -0.000399         |
|   value_loss            |    66.8              |
|-----|-----|
```

Fonte: Elaborado pelo Autor, 2024.

A melhoria na recompensa média de -992.26 para 249.79 é indicativa de que o agente conseguiu aprender uma política para controlar o módulo lunar. A alta variação na recompensa inicial sugere que, sem treinamento, o desempenho do agente era inconsistente, possivelmente resultando em tentativas de pouso desastrosas. Com o treinamento, o agente não apenas melhorou seu desempenho médio, mas também reduziu a variação nos resultados, indicando uma maior consistência nas suas ações.

Além disso, o fato de o agente ter alcançado uma recompensa positiva substancial após o treinamento indica que ele conseguiu dominar as ações necessárias para o pouso controlado, como ajustar o impulso para contrabalançar a gravidade e manobrar lateralmente para alinhar o módulo com a zona de pouso.

Esses resultados mostram que o agente, ao longo do tempo e através de um treinamento, foi capaz de aprender a navegar no ambiente e realizar um pouso lunar.



Figura 4 – Video Lunar Lander (Clique aqui)

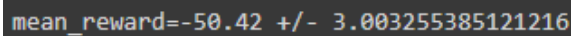
Fonte: Elaborado pelo Autor, 2024.

### 3.2 Car Racing

O objetivo do treinamento era ensinar o agente a controlar um carro em uma pista, maximizando a recompensa acumulada ao longo do episódio.

Antes do treinamento, o agente foi avaliado em 5 episódios, resultando em uma recompensa média de -50.42 com uma variação de  $\pm 3.00$ . Esse valor negativo indica que, inicialmente, o agente estava tendo dificuldades para navegar na pista, muitas vezes saindo da pista, resultando em uma baixa recompensa acumulada. A baixa variação também sugere que o comportamento do agente era consistentemente ruim, refletindo sua falta de conhecimento do ambiente.

Figura 5 – Recompensa Inicial do Car Racing

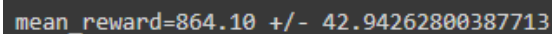


```
mean_reward=-50.42 +/- 3.003255385121216
```

Fonte: Elaborado pelo Autor, 2024.

Após o treinamento, que consistiu em uma execução de 400.000 timesteps, o agente foi novamente avaliado, apresentando uma recompensa média final de 864.10 com uma variação de  $\pm 42.942$ . Essa recompensa positiva substancial demonstra que o agente conseguiu aprender a controlar o carro, visitando as áreas da pista em menor quantidade de frames. Visto que a recompensa é de -0,1 a cada quadro e  $+1000/N$  para cada bloco da pista visitado, onde  $N$  é o número total de blocos visitados na pista. Por exemplo, se você terminou em 732 quadros, sua recompensa será  $1000 - 0,1 \cdot 732 = 926,8$  pontos. (KLIMOV, 2022)

Figura 6 – Recompensa Final do Car Racing



```
mean_reward=864.10 +/- 42.94262800387713
```

Fonte: Elaborado pelo Autor, 2024.

A maior variação na recompensa final pode indicar que o agente desenvolveu diferentes estratégias que, embora eficazes, resultam em uma pequena diferença no desempenho.

A transição de uma recompensa média de -50.42 para 864.10 reflete uma melhora notável na habilidade do agente em navegar no ambiente. Inicialmente, o agente estava realizando movimentos praticamente aleatórios, resultando em falhas consistentes em manter o carro na pista. Com o progresso do treinamento, o agente foi capaz de explorar diferentes estratégias de condução, aprendendo a acelerar, frear e virar de forma a maximizar sua eficiência na pista.

O valor da recompensa final sugere que o agente não apenas conseguiu manter o carro na pista, mas também otimizou sua trajetória para completar a volta no menor tempo possível.

A variação mais alta na recompensa final, em comparação com o desempenho inicial, também é um ponto de interesse. Ela sugere que, enquanto o agente aprendeu a realizar a tarefa, existem múltiplas trajetórias que podem ser adotadas para atingir o mesmo nível de sucesso, resultando em uma maior diversidade nos resultados.

Esses resultados demonstram que o aprendizado por reforço, quando aplicado com a política e os parâmetros corretos, pode levar a uma melhora no desempenho do agente em ambientes como o Car Racing. O agente passou de um comportamento quase aleatório e ineficaz para uma política que maximiza a recompensa acumulada, mostrando a capacidade do modelo de aprender.

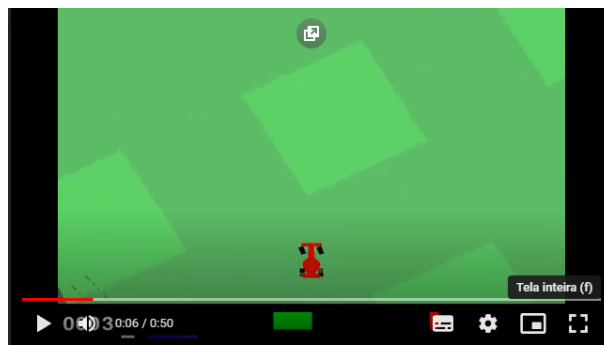


Figura 7 – Video Car Racing 25 Mil steps (Clique aqui)

Fonte: Elaborado pelo Autor, 2024.

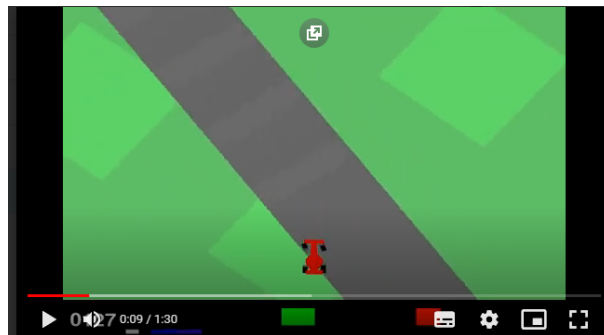


Figura 8 – Video Car Racing 50 Mil steps (Clique aqui)

Fonte: Elaborado pelo Autor, 2024.



Figura 9 – Video Car Racing 100 Mil steps (Clique aqui)

Fonte: Elaborado pelo Autor, 2024.

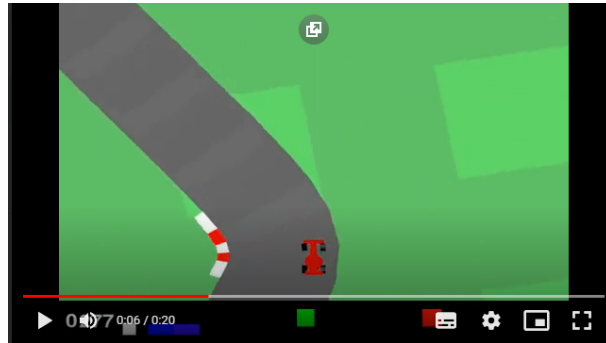


Figura 10 – Video Car Racing 200 Mil steps (Clique [aqui](#))

Fonte: Elaborado pelo Autor, 2024.

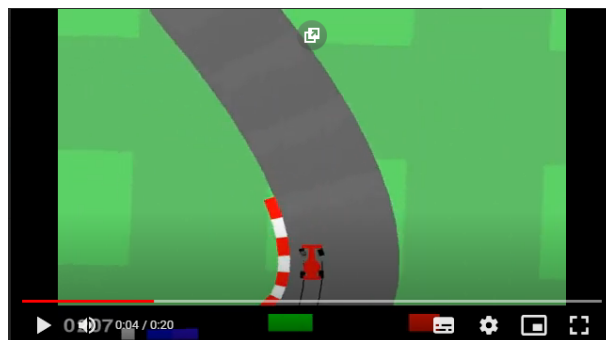


Figura 11 – Video Car Racing 400 Mil steps (Clique [aqui](#))

Fonte: Elaborado pelo Autor, 2024.



## 4 Algoritmos

### 4.1 Lunar Lander

```

1 import gymnasium as gym
2 import numpy as np
3
4 from stable_baselines3 import PPO
5
6 model = PPO(
7     "MlpPolicy",
8     "LunarLander-v2",
9     verbose=1,
10    learning_rate=3e-4,
11    n_steps=2048,
12    batch_size=64,
13    n_epochs=10,
14    gamma=0.99,
15    gae_lambda=0.95,
16    clip_range=0.2,
17    ent_coef=0.0,
18 )
19
20 from stable_baselines3.common.evaluation import evaluate_policy
21
22 eval_env = gym.make("LunarLander-v2")
23
24 mean_reward, std_reward = evaluate_policy(
25     model,
26     eval_env,
27     n_eval_episodes=10,
28     deterministic=True,
29 )
30
31 print(f"mean_reward={mean_reward:.2f} +/- {std_reward}")
32
33 #del model
34
35 model.learn(total_timesteps=int(1e6))
36
37 model.save("dqn_lunar")
38
39 mean_reward, std_reward = evaluate_policy(model, eval_env, n_eval_episodes
    =100, deterministic=True)
40
41 print(f"mean_reward={mean_reward:.2f} +/- {std_reward}")
42
43 import os

```

```

44 os.system("Xvfb:1 -screen 0 1024x768x24 &")
45 os.environ['DISPLAY'] = ':1'
46
47 import base64
48 from pathlib import Path
49
50 from IPython import display as ipythondisplay
51
52
53 def show_videos(video_path="", prefix=""):
54     """
55     Taken from https://github.com/eleurent/highway-env
56
57     :param video_path: (str) Path to the folder containing videos
58     :param prefix: (str) Filter the video, showing only the only starting
59         with this prefix
60     """
61     html = []
62     for mp4 in Path(video_path).glob("{}*.mp4".format(prefix)):
63         video_b64 = base64.b64encode(mp4.read_bytes())
64         html.append(
65             """<video alt="{}" autoplay
66                 loop controls style="height: 400px;">
67                 <source src="data:video/mp4;base64,{}" type="video/mp4"
68                     />
69             </video>""".format(
70                 mp4, video_b64.decode("ascii")
71             )
72         )
73     ipythondisplay.display(ipythondisplay.HTML(data="<br>".join(html)))
74
75 from stable_baselines3.common.vec_env import VecVideoRecorder, DummyVecEnv
76
77 def record_video(env_id, model, video_length=500, prefix="", video_folder="
78     videos/"):
79     """
80     :param env_id: (str)
81     :param model: (RL model)
82     :param video_length: (int)
83     :param prefix: (str)
84     :param video_folder: (str)
85     """
86     eval_env = DummyVecEnv([lambda: gym.make("LunarLander-v2", render_mode=
87         "rgb_array")])
88
89     eval_env = VecVideoRecorder(
90         eval_env,

```

```
88         video_folder=video_folder ,
89         record_video_trigger=lambda step: step == 0,
90         video_length=video_length ,
91         name_prefix=prefix ,
92     )
93
94     obs = eval_env.reset()
95     for _ in range(video_length):
96         action, _ = model.predict(obs)
97         obs, _, _, _ = eval_env.step(action)
98
99
100     eval_env.close()
101
102     record_video("LunarLander-v2", model, video_length=1000, prefix="ppo-
103                 lunarlander")
104
105     show_videos("videos", prefix="ppo")
```

## 4.2 Car Racing

```

1 import gymnasium as gym
2 import numpy as np
3
4 from stable_baselines3 import PPO
5 from sb3_contrib import RecurrentPPO
6
7 from stable_baselines3.common.evaluation import evaluate_policy
8 from stable_baselines3.common.vec_env import VecVideoRecorder, DummyVecEnv
9
10 env = gym.make("CarRacing-v2", render_mode="rgb_array")
11
12 model = PPO('CnnPolicy', env, verbose=1, tensorboard_log="log",
13 )
14
15 model = PPO(
16     "CnnPolicy",
17     env,
18     verbose=1,
19     learning_rate=3e-4,
20     n_steps=2048,
21     batch_size=64,
22     n_epochs=10,
23     gamma=0.99,
24     gae_lambda=0.95,
25     clip_range=0.2,
26     ent_coef=0.01,
27 )
28
29 eval_env = gym.make("CarRacing-v2", render_mode="rgb_array")
30
31 mean_reward, std_reward = evaluate_policy(
32     model,
33     eval_env,
34     n_eval_episodes=5,
35     deterministic=False,
36 )
37
38 print(f"mean_reward={mean_reward:.2f} +/- {std_reward}")
39
40 model.learn(total_timesteps=int(4e5), log_interval=10, progress_bar=False)
41
42 model.save("PPO5_CarRacing_"+str(int(400000)))
43
44 del model
45
46 model = PPO.load("PPO1_CarRacing_400000", env=eval_env)

```

```

47
48 mean_reward, std_reward = evaluate_policy(model, eval_env, n_eval_episodes
    =5, deterministic=False)
49
50 print(f"mean_reward={mean_reward:.2f} +/- {std_reward}")
51
52 import os
53 os.system("Xvfb:1 -screen 0 1024x768x24 &")
54 os.environ['DISPLAY'] = ':1'
55
56 import base64
57 from pathlib import Path
58
59 from IPython import display as ipythondisplay
60
61
62 def show_videos(video_path="", prefix=""):
63     html = []
64     for mp4 in Path(video_path).glob("{}*.mp4".format(prefix)):
65         video_b64 = base64.b64encode(mp4.read_bytes())
66         html.append(
67             """<video alt("{}" autoplay
68                 loop controls style="height: 400px;">
69                 <source src="data:video/mp4;base64,{}" type="video/mp4"
70                     />
71                 </video>""".format(
72                     mp4, video_b64.decode("ascii")
73                 )
74         )
75     ipythondisplay.display(ipythondisplay.HTML(data="<br>".join(html)))
76
77 def record_video(env_id, model, video_length=1000, prefix="", video_folder=
    "videos/"):
78     """
79     :param env_id: (str)
80     :param model: (RL model)
81     :param video_length: (int)
82     :param prefix: (str)
83     :param video_folder: (str)
84     """
85
86     eval_env = DummyVecEnv([lambda: gym.make("CarRacing-v2", render_mode="
        rgb_array")])
87
88     eval_env = VecVideoRecorder(
89         eval_env,
90         video_folder=video_folder,
91         record_video_trigger=lambda step: step == 0,
92         video_length=video_length,

```

```
91         name_prefix=prefix ,
92     )
93
94     obs = eval_env.reset()
95     for _ in range(video_length):
96         action, _ = model.predict(obs)
97         obs, _, _, _ = eval_env.step(action)
98
99
100     eval_env.close()
101
102 record_video("CarRacing-v2", model, video_length=1000, prefix="ppo2-
103             carracing")
104 show_videos("videos", prefix="ppo2")
```

## REFERÊNCIAS

- ARULKUMARAN, K.; CULLY, A.; TOGELIUS, J. Alphastar: An evolutionary computation perspective. In: *Proceedings of the genetic and evolutionary computation conference companion*. [S.l.: s.n.], 2019. p. 314–315.
- KLIMOV, O. *Car Racing*. 2022. Disponível em: <[https://www.gymlibrary.dev/environments/box2d/car\\_racing/](https://www.gymlibrary.dev/environments/box2d/car_racing/)>. Acesso em: 26 de agosto de 2024.
- RUSSELL, S. J.; NORVIG, P. *Artificial intelligence: a modern approach. Third Edition*. [S.l.]: Pearson Education, Inc., 2016.